



## PATENT ABSTRACTS OF JAPAN

(11) Publication number: **05324277 A**(43) Date of publication of application: **07.12.93**

(51) Int. Cl.

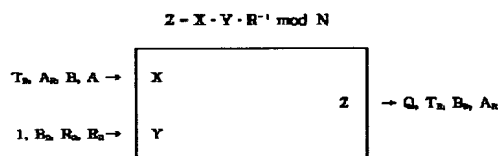
**G06F 7/72****H04L 9/06****H04L 9/14**(21) Application number: **04124982**(71) Applicant: **CANON INC**(22) Date of filing: **18.05.92**(72) Inventor: **IWAMURA KEIICHI  
YAMAMOTO TAKAHISA**(54) **CODE COMMUNICATION METHOD**

COPYRIGHT: (C)1993,JPO&amp;Japio

(57) Abstract:

**PURPOSE:** To provide the circuit executing the power residue arithmetic operation and the residue multiplication at high speed with smaller circuit by repeatedly executing the residue multiplication using both modulo N and prime R of the residue.

**CONSTITUTION:** In the residue arithmetic circuit, the outputs for input pairs (A, RR), (B, RR), (AR, BR) (TR, 1) are AR, BR, TR, and Q. In this case, the power residue arithmetic operation and the residue multiplication are executed by repeating the operation of  $Z = X \cdot Y \cdot R^{-1} \bmod N$ . Therefore, the required arithmetic operation is executed by the same or similar type arithmetic circuit. In performing the arithmetic operation with the use of the Montgomery residue multiplication  $Z = X \cdot Y \cdot R^{-1} \bmod N = (X \cdot Y + S \cdot N) / R$ , in this case,  $S = X \cdot Y \cdot N^{-1} \bmod N$ , the residue multiplication and the power residue arithmetic operation can be executed while simply repeating the Montgomery residue multiplication by using the input value satisfying the condition.



(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平5-324277

(43) 公開日 平成5年(1993)12月7日

(51) Int. Cl. <sup>5</sup>	識別記号	庁内整理番号	F I	技術表示箇所
G 0 6 F 7/72		9188-5B		
H 0 4 L 9/06				
9/14				
		7117-5K	H 0 4 L 9/02	Z

審査請求 未請求 請求項の数10(全 20 頁)

(21) 出願番号 特願平4-124982

(22) 出願日 平成4年(1992)5月18日

(71) 出願人 000001007

キヤノン株式会社

東京都大田区下丸子3丁目30番2号

(72) 発明者 岩村 恵市

東京都大田区下丸子3丁目30番2号キヤノン株式会社内

(72) 発明者 山本 貴久

東京都大田区下丸子3丁目30番2号キヤノン株式会社内

(74) 代理人 弁理士 丸島 儀一

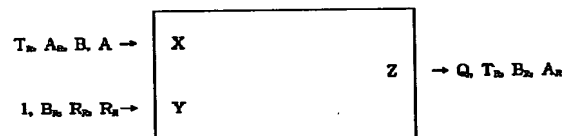
(54) 【発明の名称】 暗号通信方法

(57) 【要約】

【目的】 暗号通信方法において、必要となる剰余乗算またはべき乗剰余演算を小さな回路規模で高速に演算する。

【構成】 剰余乗算  $Q = A \cdot B \bmod N$  及びべき乗剰余演算  $C = M^e \bmod N$  を、 $N$  と素な整数  $R$  を用いた同形の演算  $Z = U \cdot V \cdot R^{-1} \bmod N$  の繰り返しにより実現する。また、この演算を同一の回路による繰り返し演算、または同一構成の複数の回路による並列処理によって実行する。

$$Z = X \cdot Y \cdot R^{-1} \bmod N$$



1

2

【特許請求の範囲】

【請求項1】  $N$ を法とする整数 $A$ 、 $B$ の剰余乗算 $Q = A \cdot B \bmod N$ を利用して、通信内容の暗号化または復号を行なう暗号通信方法において、

入力データ $U$ 、 $V$ に対して、 $N$ と素である整数 $R$ を用いて、 $Z = U \cdot V \cdot R^{-1} \bmod N$ を演算して出力する演算部を1つ以上設け、

当該演算部に対して、

$A$ と、 $R_1 = R^2 \bmod N$ なる定数 $R_1$ とを入力して、 $A_1 = A \cdot R_1 \cdot R^{-1} \bmod N$ を出力させ、

$B$ と、前記定数 $R_1$ とを入力して、 $B_1 = B \cdot R_1 \cdot R^{-1} \bmod N$ を出力させ、

出力された前記 $A_1$ と前記 $B_1$ とを入力して、 $T_1 = A_1 \cdot B_1 \cdot R^{-1} \bmod N$ を出力させ、

出力された前記 $T_1$ と定数1とを入力して、 $T_1 \cdot 1 \cdot R^{-1} \bmod N$ を $Q$ として出力させることにより、前記剰余乗算 $Q = A \cdot B \bmod N$ を実行することを特徴とする暗号通信方法。

【請求項2】  $N$ を法とする整数 $M$ 、 $e$ に関するべき乗剰余演算： $C = M^e \bmod N$ を利用して、通信内容の暗号化または復号を行なう暗号通信方法において、

入力データ $U$ 、 $V$ に対して、 $N$ と素である整数 $R$ を用いて、 $Z = U \cdot V \cdot R^{-1} \bmod N$ を演算して出力する演算部を1つ以上設け、

当該演算部に対して、 $M$ と、 $R_1 = R^2 \bmod N$ なる定数 $R_1$ とを入力して、

$M_1 = M \cdot R_1 \cdot R^{-1} \bmod N$ を出力させ、

$e$ の2進表現を $e = [e^1, e^{1-1}, \dots, e^1]$ とし、 $C_1$ の初期値を $C_1 = R_1 \cdot R^{-1} \bmod N$ として、順次高位ビットからの $e^i$ の値に従って、 $e^i = 1$ なるときに、前記演算部に対して $C_1$ と $M_1$ とを入力して、 $C_1 \cdot M_1 \cdot R^{-1} \bmod N$ を新たな $C_1$ として出力させ、

更に、前記 $e^i$ における $i$ が1より大なるときには、前記演算部に対して2つの入力データとして共に $C_1$ を入力して、 $C_1 \cdot C_1 \cdot R^{-1} \bmod N$ を新たな $C_1$ として出力させ、

全ての前記 $e^i$ に対する処理の終了後に、前記演算部に対して $C_1$ と定数1とを入力して、 $C = C_1 \cdot 1 \cdot R^{-1} \bmod N$ を出力させることにより、前記べき乗剰余演算 $C = M^e \bmod N$ を実行することを特徴とする暗号通信方法。

【請求項3】  $N$ を法とする整数 $M$ 、 $e$ に関するべき乗剰余演算： $C = M^e \bmod N$ を利用して、通信内容の暗号化または復号を行なう暗号通信方法において、

入力データ $U$ 、 $V$ に対して、 $N$ と素である整数 $R$ を用いて、 $Z = U \cdot V \cdot R^{-1} \bmod N$ を演算して出力する演算部を1つ以上設け、

当該演算部に対して、 $M$ と、 $R_1 = R^2 \bmod N$ なる定数 $R_1$ とを入力して、

$M_1 = M \cdot R_1 \cdot R^{-1} \bmod N$ を出力させ、

$e$ の2進表現を $e = [e^1, e^{1-1}, \dots, e^1]$ とし、 $C_1$ の初期値を $C_1 = R_1 \cdot R^{-1} \bmod N$ として、順次低位ビットからの $e^i$ の値に従って、 $e^i = 1$ なるときに、前記演算部に対して $C_1$ と $M_1$ とを入力して、 $C_1 \cdot M_1 \cdot R^{-1} \bmod N$ を新たな $C_1$ として出力させ、

更に、前記 $e^i$ における $i$ が1より小なるときには、前記演算部に対して2つの入力データとして共に $M_1$ を入力して、 $M_1 \cdot M_1 \cdot R^{-1} \bmod N$ を新たな $M_1$ として出力させ、

10 全ての前記 $e^i$ に対する処理の終了後に、前記演算部に対して $C_1$ と定数1とを入力して、 $C = C_1 \cdot 1 \cdot R^{-1} \bmod N$ を出力させることにより、前記べき乗剰余演算 $C = M^e \bmod N$ を実行することを特徴とする暗号通信方法。

【請求項4】 前記演算部に、定数 $R_1$ と定数1とを入力して、出力 $R_1 \cdot 1 \cdot R^{-1} \bmod N$ を $C_1$ の初期値とすることを特徴とする請求項2あるいは3に記載の暗号通信システム。

【請求項5】  $n$ を $N < 2^n$ なる値とするとき、前記演算部において、定数 $R$ 、入力データ $U$ 、 $V$ が、 $u = 1$ かつ $r > 1$ 、または、 $u > 1$ かつ $r = u + 1$ なる $u$ 、 $r$ に対して、 $R = 2^{u+r}$ 、 $U < 2^{u+r}$ 、 $V < 2^{u+r}$ を満たしていることを特徴とする請求項1ないし4に記載の暗号通信システム。

【請求項6】 入力された整数 $A$ 、 $B$ に対する $N$ を法とした剰余乗算 $Q = A \cdot B \bmod N$ を利用して、通信内容の暗号化または復号を行なう暗号通信方法において、 $N$ と素である整数 $R$ を用いて、

入力された $A$ 及び前記 $R$ より $A \cdot R \bmod N$ を演算してその結果を $A_1$ とし、

入力された $B$ 及び前記 $R$ より $B \cdot R \bmod N$ を演算してその結果を $B_1$ とし、

前記演算結果 $A_1$ 、 $B_1$ 及び前記 $R$ に基づき、 $A_1 \cdot B_1 \cdot R^{-1} \bmod N$ を求めてその結果を $T_1$ とし、

前記 $T_1$ と前記 $R$ とにより $T_1 \cdot R^{-1} \bmod N$ を演算し、その結果として $Q$ を求めるようにし、

前記 $T_1$ を求める演算を、 $A_1$ を任意の整数 $v$ による前記 $A_1$ の $v$ ビット毎の分割、 $Y = 2^v$ として、

$T_1 = (T_{1-1} + A_1 \cdot B_1 \cdot Y + M_{1-1} \cdot N) / Y$

40  $M_{1-1} = (T_{1-1} \bmod Y) \cdot (-N^{-1} \bmod Y) \bmod Y$ の順次演算により実行することを特徴とする暗号通信方法。

【請求項7】 前記順次演算における各1回の演算を、1つの演算素子によって実行し、前記順次演算全体をパイプライン処理により実行することを特徴とした請求項6に記載の暗号通信方法。

【請求項8】 前記順次演算において、 $Y$ による乗算または除算を、加算において、ビット位置をずらして加算することにより実行することを特徴とした請求項6に記載の暗号通信方法。

3

【請求項9】 前記順次演算において、 $B_i$ 、 $N_i$ をそれぞれ任意の整数 $d$ による前記 $B_i$ 、 $N$ の $d$ ビット毎の分割として、 $A_i \cdot B_{i-1}$ と $M_{i-1} \cdot N_i$ とを演算し、該演算結果と前回の順次演算結果 $T_{i-1}$ を加算することを特徴とした請求項6に記載の暗号通信方法。

【請求項10】 入力された整数 $A$ 、 $B$ に対する $N$ を法とした剰余乗算 $Q = A \cdot B \bmod N$ を利用して、通信内容の暗号化または復号を行なう暗号通信方法において、 $N$ と素である整数 $R$ を用いて、

入力された $A$ 及び前記 $R$ より $A \cdot R \bmod N$ を演算して\*10

$$T_i = (T_{i-1} / Y + A_i \cdot B_i) + M_i \cdot N$$

$$M_{i-1} = ((T_{i-1} / Y + A_i \cdot B_i) \bmod Y) \cdot (-N^{-1} \bmod Y) \bmod Y$$

の順次演算により実行することを特徴とする暗号通信方法。

【発明の詳細な説明】

【0001】

【産業上の利用分野】本発明はコンピュータネットワークにおけるホームバンク、ファームバンク、電子メール及び電子会議などの様々な通信サービスに用いられる暗号化技術に関する。特にべき乗剰余演算及び剰余乗算を用いる暗号方式（RSA暗号、エルガマル暗号等）、鍵共有方式（DH型鍵共有方式、ID-based鍵共有方式等）、零知識証明方式等を用いて暗号通信を行うシステムに関する。

【0002】

【従来の技術】近年、コンピュータネットワークを用いた情報通信システムの急速な進展とともに、データ内容の保護を目的とする暗号技術の重要性が高まっている。特に、コンピュータネットワークの高速化・大容量化が進展する中で、高速な暗号技術が不可欠になりつつある。

【0003】かかる暗号技術において、べき乗剰余演算及び剰余乗算は種々の暗号技術に用いられている重要な演算であり、次のような利用例を挙げることができる。

【0004】まず、暗号方式には秘密鍵暗号方式と公開鍵暗号方式があることが知られている。公開鍵暗号方式では暗号化鍵と復号鍵とが異なり、暗号化鍵は公開し、復号鍵は受信者が秘密に保持するもので、公開された暗号化鍵から復号鍵を推定するのが困難なようになっているものである。その公開鍵暗号方式としてRSA暗号やエルガマル暗号などのべき乗剰余演算及び剰余乗算に基づく暗号がよく用いられている。更にこれらの暗号は、秘密通信機能の他に認証と呼ばれるもう1つの用途があることが注目されている。認証とは、通信文の送信者が正しいかどうかを検査する機能であり、デジタル署名とも呼ばれている。これらの暗号を用いたデジタル署名では、送信者のみが知っている秘密鍵で署名でき、偽造できないので安全であり認証通信として金融機関などで多く用いられている。

【0005】また、同一の鍵を送信者と受信者が秘密に

4

\*その結果を $A_i$ とし、

入力された $B$ 及び前記 $R$ より $B \cdot R \bmod N$ を演算してその結果を $B_i$ とし、

前記演算結果 $A_i$ 、 $B_i$ 及び前記 $R$ に基づき、 $A_i \cdot B_i \cdot R^{-1} \bmod N$ を求めてその結果を $T_i$ とし、

前記 $T_i$ と前記 $R$ とにより $T_i \cdot R^{-1} \bmod N$ を演算し、その結果として $Q$ を求めるようにし、

前記 $T_i$ を求める演算を、 $A_i$ を任意の整数 $v$ による前記 $A_i$ の $v$ ビット毎の分割、 $Y = 2^v$ として、

共有する秘密鍵暗号方式として、乱数をデータに加えるバーナム暗号が知られているが、その乱数として平方剰余と呼ばれるべき乗剰余演算及び剰余乗算に基づく乱数が知られている。

【0006】また、以上の秘密鍵暗号方式及び公開鍵暗号方式は、鍵配送方式または鍵共有方式と呼ばれる技術とともに用いられることが多い。鍵配送方式としては、DiffieとHellmanによるDH型鍵配送方式がよく知られているが、この方式もべき乗剰余演算及び剰余乗算を用いて演算を行う。さらに、鍵共有方式としてID-based鍵共有方式が注目されているが、この方式を含む種々の鍵共有方式においてべき乗剰余演算及び剰余乗算が用いられている。

【0007】他に、暗号技術には零知識証明と呼ばれるものがある。これは自分がある知識を持っていることを、その内容をいっさい告げることなく（＝零知識）、相手に納得させる（＝証明）方法である。これにも、べき乗剰余演算及び剰余乗算に基づく種々の手法がある。

【0008】以上の暗号技術の詳細については池野信一、小山謙二著“現代暗号理論”、電子情報通信学会（1986）及び辻井重男、笠原正雄著“暗号と情報セキュリティ”、昭晃堂（1990）等に詳しく説明されている。

【0009】従って、種々の暗号システムを効率よく構成するために、効率的なべき乗剰余演算及び剰余乗算回路の実現が望まれていた。更に、高速なべき乗剰余演算及び剰余乗算回路が構成できれば、種々の暗号システムの高速化が実現できる。

【0010】ところで、 $N$ を法とする剰余乗算を演算する方法として、 $N$ と素な整数 $R$ を用いて演算を行う手法がある。例えば、モンゴメリーによって提案された手法（モンゴメリー法）（Montgomery, P.L.: “Modular multiplication without trial division,” Math. of Computation, Vol. 44, 1985, pp. 519-521）は、 $Q = A \cdot B \bmod N$ の代わりに $Q = A \cdot B \cdot R^{-1} \bmod N$ を演算することで、除算を行うことなく剰余乗算を計算することができる。

【0011】一方、処理を高速化していく1つの手法として、並列処理がある。その代表的なアーキテクチャと

してシストリックアレイが知られている。シストリックアレイは処理を数種類の演算素子（プロセッシング・エレメント：以後PE）によるパイプライン処理によって実行し、高速処理を実現する。また、制御がPE単位の局所的なもので済み容易である。従って、シストリックアレイは全体構造の規則性とPE単位の局所性を有し、VLSI等の大規模な処理の装置化を容易にするアーキテクチャとして知られている。このような並列処理の手法は大規模な処理を必要とする大きな整数に対するべき乗剰余演算及び剰余乗算の高速化にも適していると考えられるが、従来の手法の中でシストリックアレイ等の並列処理の手法をべき乗剰余演算及び剰余乗算に対して適用したものは殆どなかった。

【0012】そこで、本出願人は、先に特願平3-225986号として、シストリックアレイを用いた剰余乗算回路を提案したが、これはモンゴメリー法を用いたものではない。一方、モンゴメリー法を用いたアレイがイブンによって提案されている。（Shimon Even: "Systolic modular multiplication," Advances in Cryptology-CRYPT '90, pp. 619-624, Springer-Verlag.）

【0013】

【発明が解決しようとしている課題】上述のような暗号システムに用いられるべき乗剰余演算及び剰余乗算で用いられる整数は、十分な安全性を確保するために512ビット以上のビット数を持つことが要求される。このように大きな整数に対するべき乗剰余演算及び剰余乗算を通常のコンピュータを用いて高速に演算することは困難であった。

【0014】また、モンゴメリー法を繰り返してべき乗剰余演算を実行する場合、剰余乗算を繰り返す度に出力の最大ビット数が大きくなり、同じ回路によってべき乗剰余演算を実行することは困難であった。これについて、イブンのアレイは、剰余乗算出力のビット数が入力値のビット数を越えた場合の処理を行うPEについて示されておらず、べき乗剰余演算に対しては不十分なものになっている。

【0015】さらに、従来のモンゴメリー法は後述するように $Q=A \cdot B \cdot R^{-1} \bmod N$ の演算を行う前後に、A、B及びQに対して別の演算を行う必要があり、数種類の演算手段が必要であった。

【0016】また、特に、上述のイブンのアレイは、乗算 $T=A \cdot B$ を実行するアレイと、定数として扱われるRに対する剰余演算 $Q=T \cdot R^{-1} \bmod N$ を実行するアレイから構成されている。従って、イブンのシストリックアレイは、Tを演算するアレイとQを演算するアレイが2種類必要であるために効率的ではなかった。さらに、PE内で行なれる演算として1ビット毎の演算のみを提案しており、柔軟性に欠けていた。

【0017】

【課題を解決するための手段】そこで、本発明の目的

は、上述の欠点を除去し、暗号通信におけるべき乗剰余演算及び剰余乗算を、剰余の法となるNと素であるRを用いた剰余乗算を繰り返すだけで実行する方法を提供することにある。

【0018】また、本発明の他の目的は、モンゴメリー法を用いて、より小さな回路規模で高速にべき乗剰余演算及び剰余乗算を実行する回路を実現することにある。

【0019】かかる課題を解決するために、本発明では、Nを法とする整数A、Bの剰余乗算 $Q=A \cdot B \bmod N$ を利用して、通信内容の暗号化または復号を行なう暗号通信方法において、入力データU、Vに対して、Nと素である整数Rを用いて、 $Z=U \cdot V \cdot R^{-1} \bmod N$ を演算して出力する演算部を1つ以上具える。

【0020】また、本発明の他の態様によれば、Nを法とする整数M、eに関するべき乗剰余演算： $C=M^e \bmod N$ を利用して、通信内容の暗号化または復号を行なう暗号通信方法において、入力データU、Vに対して、Nと素である整数Rを用いて、 $Z=U \cdot V \cdot R^{-1} \bmod N$ を演算して出力する演算部を1つ以上具える。

【0021】また、本発明の他の態様によれば、入力された整数A、Bに対するNを法とした剰余乗算 $Q=A \cdot B \bmod N$ を利用して、通信内容の暗号化または復号を行なう暗号通信方法において、Nと素である整数Rを用いて、入力されたA及び前記Rより $A \cdot R \bmod N$ を演算してその結果を $A_1$ とする演算工程と、入力されたB及び前記Rより $B \cdot R \bmod N$ を演算してその結果を $B_1$ とする演算工程と、前記演算結果 $A_1$ 、 $B_1$ 及び前記Rに基づき、 $A_1 \cdot B_1 \cdot R^{-1} \bmod N$ を求めてその結果を $T_1$ とする演算工程と、前記 $T_1$ と前記Rとにより $T_1 \cdot R^{-1} \bmod N$ を演算し、その結果としてQを求める演算工程とを有し、前記 $T_1$ を求める演算工程に、 $A_1$ を任意の整数vによる前記 $A_1$ のvビット毎の分割、 $Y=2^v$ として、

$$T_i = (T_{i-1} + A_1 \cdot B_1 \cdot Y + M_{i-1} \cdot N) / Y$$

$$M_{i-1} = (T_{i-1} \bmod Y) \cdot (-N^{-1} \bmod Y) \bmod Y$$

の順次演算により実行する演算工程とを具える。

【0022】また、本発明の他の態様によれば、入力された整数A、Bに対するNを法とした剰余乗算 $Q=A \cdot B \bmod N$ を利用して、通信内容の暗号化または復号を行なう暗号通信方法において、Nと素である整数Rを用いて、入力されたA及び前記Rより $A \cdot R \bmod N$ を演算してその結果を $A_1$ とする演算工程と、入力されたB及び前記Rより $B \cdot R \bmod N$ を演算してその結果を $B_1$ とする演算工程と、前記演算結果 $A_1$ 、 $B_1$ 及び前記Rに基づき、 $A_1 \cdot B_1 \cdot R^{-1} \bmod N$ を求めてその結果を $T_1$ とする演算工程と、前記 $T_1$ と前記Rとにより $T_1 \cdot R^{-1} \bmod N$ を演算し、その結果としてQを求める演算工程とを有し、前記 $T_1$ を求める演算工程に、 $A_1$ を任意の整数vによる前記 $A_1$ のvビット毎の分割、 $Y=2^v$ として、

7

$$T_i = (T_{i-1} / Y + A_i \cdot B_i) + M_i \cdot N$$

$$M_{i-1} = ((T_{i-1} / Y + A_i \cdot B_i) \bmod Y) \cdot (-N^{-1} \bmod Y) \bmod Y$$

の順次演算により実行する演算工程を具える。

【0023】

【作用】入力データU、Vに対して、Nと素である整数Rを用いて、 $Z = U \cdot V \cdot R^{-1} \bmod N$ を演算して出力する1つ以上の演算部に対して、Aと、 $R_1 = R^2 \bmod N$ なる定数 $R_1$ とを入力して、 $A_1 = A \cdot R_1 \cdot R^{-1} \bmod N$ を出力させ、Bと、前記定数 $R_1$ とを入力して、 $B_1 = B \cdot R_1 \cdot R^{-1} \bmod N$ を出力させ、出力された前記A<sub>1</sub>と前記B<sub>1</sub>とを入力して、 $T_1 = A_1 \cdot B_1 \cdot R^{-1} \bmod N$ を出力させ、出力された前記T<sub>1</sub>と定数1とを入力して、 $T_1 \cdot 1 \cdot R^{-1} \bmod N$ をQとして出力させることにより、前記剰余乗算 $Q = A \cdot B \bmod N$ を実行する。

【0024】入力データU、Vに対して、Nと素である整数Rを用いて、 $Z = U \cdot V \cdot R^{-1} \bmod N$ を演算して出力する1つ以上の演算部に対して、Mと、 $R_1 = R^2 \bmod N$ なる定数 $R_1$ とを入力して、 $M_1 = M \cdot R_1 \cdot R^{-1} \bmod N$ を出力させ、eの2進表現を $e = (e^1, e^{1-1}, \dots, e^1)$ とし、C<sub>1</sub>の初期値を $C_1 = R_1 \cdot R^{-1} \bmod N$ として、順次高位ビットからのe<sup>1</sup>の値に従って、e<sup>1</sup> = 1なるときに、前記演算部に対してC<sub>1</sub>とM<sub>1</sub>とを入力して、 $C_1 \cdot M_1 \cdot R^{-1} \bmod N$ を新たなC<sub>1</sub>として出力させ、更に、前記e<sup>1</sup>における1が1より大なるときには、前記演算部に対して2つの入力データとして共にC<sub>1</sub>とM<sub>1</sub>とを入力して、 $C_1 \cdot C_1 \cdot R^{-1} \bmod N$ を新たなC<sub>1</sub>として出力させ、全ての前記e<sup>1</sup>に対する処理の終了後に、前記演算部に対してC<sub>1</sub>と定数1とを入力して、 $C = C_1 \cdot 1 \cdot R^{-1} \bmod N$ を出力させることにより、前記べき乗剰余演算 $C = M^e \bmod N$ を実行する。

【0025】入力データU、Vに対して、Nと素である整数Rを用いて、 $Z = U \cdot V \cdot R^{-1} \bmod N$ を演算して出力する1つ以上の演算部に対して、Mと、 $R_1 = R^2 \bmod N$ なる定数 $R_1$ とを入力して、 $M_1 = M \cdot R_1 \cdot R^{-1} \bmod N$ を出力させ、eの2進表現を $e = (e^1, e^{1-1}, \dots, e^1)$ とし、C<sub>1</sub>の初期値を $C_1 = R_1 \cdot R^{-1} \bmod N$ として、順次低位ビットからのe<sup>1</sup>の値に従って、e<sup>1</sup> = \*

$$T_i = (T_{i-1} / Y + A_i \cdot B_i) + M_i \cdot N$$

$$M_{i-1} = ((T_{i-1} / Y + A_i \cdot B_i) \bmod Y) \cdot (-N^{-1} \bmod Y) \bmod Y$$

の順次演算により実行する。

【0028】

【実施例】以下、Nを法とする剰余乗算を、Nと素である整数Rを用いた値に対する剰余乗算として、モンゴメリーによって提案された手法（モンゴメリー法）を例にとり説明を行う。まず、べき乗剰余演算及び剰余乗算を用いる暗号システムについて示し、次にモンゴメリー法を用いたべき乗剰余演算及び剰余乗算を行う場合の前後の処理法とモンゴメリー法を用いた剰余乗算の入出力の整合性について示す。さらに、モンゴメリー法を実行するPEを示し、それを複数並列に用いることによってべ

8

\* 1なるときに、前記演算部に対してC<sub>1</sub>とM<sub>1</sub>とを入力して、 $C_1 \cdot M_1 \cdot R^{-1} \bmod N$ を新たなC<sub>1</sub>として出力させ、更に、前記e<sup>1</sup>における1がtより小なるときには、前記演算部に対して2つの入力データとして共にM<sub>1</sub>とM<sub>1</sub>とを入力して、 $M_1 \cdot M_1 \cdot R^{-1} \bmod N$ を新たなM<sub>1</sub>として出力させ、全ての前記e<sup>1</sup>に対する処理の終了後に、前記演算部に対してC<sub>1</sub>と定数1とを入力して、 $C = C_1 \cdot 1 \cdot R^{-1} \bmod N$ を出力させることにより、前記べき乗剰余演算 $C = M^e \bmod N$ を実行する。

【0026】入力された整数A、Bに対するNを法とした剰余乗算 $Q = A \cdot B \bmod N$ を、Nと素である整数Rを用いて、入力されたA及び前記Rより $A \cdot R \bmod N$ を演算してその結果をA<sub>1</sub>とし、入力されたB及び前記Rより $B \cdot R \bmod N$ を演算してその結果をB<sub>1</sub>とし、前記演算結果A<sub>1</sub>、B<sub>1</sub>及び前記Rに基づき、 $A_1 \cdot B_1 \cdot R^{-1} \bmod N$ を求めてその結果をT<sub>1</sub>とし、前記T<sub>1</sub>と前記Rとにより $T_1 \cdot R^{-1} \bmod N$ を演算し、その結果としてQを求めるようにし、前記T<sub>1</sub>を求める演算を、A<sub>1</sub>を任意の整数vによる前記A<sub>1</sub>のvビット毎の分割、 $Y = 2^v$ として、

$$T_i = (T_{i-1} + A_i \cdot B_i \cdot Y + M_{i-1} \cdot N) / Y$$

$$M_{i-1} = (T_{i-1} \bmod Y) \cdot (-N^{-1} \bmod Y) \bmod Y$$

の順次演算により実行する。

【0027】入力された整数A、Bに対するNを法とした剰余乗算 $Q = A \cdot B \bmod N$ を、Nと素である整数Rを用いて、入力されたA及び前記Rより $A \cdot R \bmod N$ を演算してその結果をA<sub>1</sub>とし、入力されたB及び前記Rより $B \cdot R \bmod N$ を演算してその結果をB<sub>1</sub>とし、前記演算結果A<sub>1</sub>、B<sub>1</sub>及び前記Rに基づき、 $A_1 \cdot B_1 \cdot R^{-1} \bmod N$ を求めてその結果をT<sub>1</sub>とし、前記T<sub>1</sub>と前記Rとにより $T_1 \cdot R^{-1} \bmod N$ を演算し、その結果としてQを求めるようにし、前記T<sub>1</sub>を求める演算を、A<sub>1</sub>を任意の整数vによる前記A<sub>1</sub>のvビット毎の分割、 $Y = 2^v$ として、

べき乗剰余演算及び剰余乗算を効率的に実行する回路を示す。

【0029】【暗号システム】まず、図1に示すn対nの通信系における暗号システムについて説明する。図1における結線はローカルエリアネットワーク（LAN）のような局所的な通信網、または電話回線のような大域的な通信網を表す。ここで、A～Zは利用者であり、それぞれに通信網につながるための通信機（通信端末）Tが割り当てられている。暗号装置は、入力された情報を暗号化して出力するものであり、例えば、その通信機Tに内蔵させ、通信機Tが暗号化情報を出力する構成とし

てもよいし、通信機Tと通信網の間に挿入させて、通信機Tの出力を暗号化して通信網に出力するようにしてもよい。また、通信機に接続され、通信機に情報を出力する装置に内蔵させることもできる。また、暗号装置が通信機に常時接続されていなくても、ICカードのような携帯用の装置に暗号装置を内蔵し、必要なときに通信機または通信機に接続された装置と接続する構成としてもよい。このような暗号装置によって、秘密通信や認証通信及び鍵共有、零知識証明などの暗号通信を行うことができる。

【0030】このような暗号装置で必要となる、剰余乗算回路またはべき乗剰余演算回路の例としては、平文Mを入力し、e及びNを他の入力または記憶された値として、暗号 $C=M^e \bmod N$ を出力するべき乗剰余演算回路を考えることができる。この場合、べき乗剰余演算回路が暗号装置そのものとなる。秘密通信の場合、同様の暗号装置により、逆演算 $M=C^d \bmod N$ によって復号を行\*

$$(T+M \cdot N) / R = T \cdot R^{-1} \bmod N \quad (1)$$

証明：略

従って、剰余乗算： $Q=A \cdot B \bmod N$ を実行する場合、Nに対して素である整数Rを用いて次のようにして※

$$A_1 = A \cdot R \bmod N \quad (2)$$

$$B_1 = B \cdot R \bmod N \quad (3)$$

$$T = A_1 \cdot B_1 \quad (4)$$

$$T_1 = T \cdot R^{-1} \bmod N = (T+M \cdot N) / R \quad (5)$$

$$Q = T_1 \cdot R^{-1} \bmod N \quad (6)$$

ここで、式(4)、(5)の演算をモンゴメリーの剰余乗算と呼ぶとすると、モンゴメリーの剰余乗算は次のよう★

$$T_1 = A_1 \cdot B_1 \cdot R^{-1} \bmod N \\ = (A_1 \cdot B_1 + M \cdot N) / R \quad (7)$$

$$\text{ただし、} M = A_1 \cdot B_1 \cdot N' \bmod R \quad (8)$$

モンゴメリーの剰余乗算においてNが奇数の場合、 $R=2^n$  (nは任意の整数)と選べば、RはNに対して素な整数になる。この場合、Rによる除算はビットシフトのみで済むので、式(5)または式(7)のモンゴメリーの剰余乗算は乗算のみによって実行できる。

【0037】このとき、式(2)、(3)及び(6)の前後の処理もまたモンゴメリーの剰余乗算によって次のように実行できる。

【0038】

$$A_1 = A \cdot R \bmod N = A \cdot R_1 \cdot R^{-1} \bmod N$$

$$B_1 = B \cdot R \bmod N = B \cdot R_1 \cdot R^{-1} \bmod N$$

$$Q = T_1 \cdot R^{-1} \bmod N = T_1 \cdot 1 \cdot R^{-1} \bmod N$$

ただし、 $R_1 = R^2 \bmod N$

INPUT M, e, N, R<sub>1</sub>

$$M_1 = M \cdot R_1 \cdot R^{-1} \bmod N \quad (9)$$

$$C_1 = 1 \cdot R_1 \cdot R^{-1} \bmod N \quad (10)$$

FOR i=t TO 1

$$\text{IF } e_i = 1 \text{ THEN } C_1 = C_1 \cdot M_1 \cdot R^{-1} \bmod N \quad (11)$$

$$\text{IF } i > 1 \text{ THEN } C_1 = C_1 \cdot C_1 \cdot R^{-1} \bmod N \quad (12)$$

\*うこともある。また、剰余乗算回路またはべき乗剰余演算回路を暗号装置の一部として、暗号装置への外部からの入力あるいは暗号装置内の他の処理部の処理結果をこの回路に入力して、演算を実行し、演算結果を暗号装置の外部への出力あるいは暗号装置内の他の処理部に対する入力とする構成をとることもできる。

【0031】また、記憶媒体へのアクセスを通信と見なした場合は、磁気ディスク等のような記憶媒体と、この記録媒体へのアクセス装置が通信機に相当し、通信系と同様に記憶系においても本発明による回路を用いた暗号装置によって、暗号システムを利用することができる。

【0032】〔モンゴメリーの剰余乗算〕次の定理がモンゴメリーによって導かれた。

【0033】定理1：NとRを互いに素な整数、Tを任意の整数とし、 $N' = -N^{-1} \bmod R$ とし、 $M = T \cdot N' \bmod R$ とすると、次の関係を満足する。

【0034】

※行うことができる。

【0035】

★うに表すことができる。

【0036】

R<sub>1</sub>はNによって一意に定まる値であるので、Nを定めたときに定まり、定数として扱うことができる。従って、図2に示すように $Z = X \cdot Y \cdot R^{-1} \bmod N$ を実行する演算回路を用いて、式(2)～(6)の演算が共通に実行でき、求める剰余乗算： $Q = A \cdot B \bmod N$ が演算される。図2は入力組(A, R<sub>1</sub>), (B, R<sub>1</sub>), (A<sub>1</sub>, B<sub>1</sub>), (T<sub>1</sub>, 1)に対する出力が各々A<sub>1</sub>, B<sub>1</sub>, T<sub>1</sub>, Qであることを示している。

40 【0039】〔モンゴメリーのべき乗剰余演算1〕また、モンゴメリー法を用いて、べき乗剰余演算： $C = M^e \bmod N$ も次のようにして実行される。

【0040】

11

12

NEXT

 $C = C_1 \cdot 1 \cdot R^{-1} \bmod N$ 

(13)

従って、べき乗剰余演算もモンゴメリーの剰余乗算のみによって実行できる。なお、式(10)に示す $C_1$ の初期値は、 $R_1$ と $N$ によって定まるので定数として扱うこともできる。以後、このようにモンゴメリーの剰余乗算のみを用いたべき乗剰余演算をモンゴメリーのべき乗剰余演算と呼ぶ。

【0041】ここで、以上のように、モンゴメリーのべき乗剰余演算を実行する場合、1つの演算結果を次の演算の入力として乗算を繰り返すので、各乗算を同一の回路構成で実現しようとするとき、出力の最大ビット数が10 入力の最大ビット数を越えてしまうと実現が困難となる。

【0042】そこで、式(7)のモンゴメリーの剰余乗算において、入力と出力の最大ビット数が等しくなるための条件を以下に考察する。

【0043】定理2：式(7)、(8)において $A_1 < 2^{u+r}$ 、 $B_1 < 2^{u+r}$ 、 $N < 2^u$ 、 $R = 2^{u+r}$ としたとき、 $T_1 < 2^{u+r}$ となるためには、 $u=1$ かつ $r>1$ 、20 または、 $u>1$ かつ $r=u+1$ ならば十分である。

【0044】証明：

$R = 2^{u+r}$ とすると式(8)より $M < 2^{u+r}$ 。

$A_1 < 2^{u+r}$ 、 $B_1 < 2^{u+r}$ 、 $N < 2^u$ とすると、

$A_1 \cdot B_1 < 2^{2(u+r)}$ 、 $M \cdot N < 2^{u+r+u}$ 。

キャリーによる桁上りを考慮して、

$A_1 \cdot B_1 + M \cdot N < \max(2^{2(u+r)+1}, 2^{2(u+r)+1})$ 。

よって、 $T_1 < \max(2^{2(u+r)+1-r}, 2^{u+1})$ 。

従って、 $2^{u+2u+1-r} \leq 2^{u+1}$ の場合： $T_1 < 2^{u+1}$ 。

$\therefore u=1, r>1$  (14) 30

$2^{u+2u+1-r} > 2^{u+1}$ の場合： $T_1 < 2^{u+2u+1-r}$ 。

$\therefore u>1, r=u+1$  (15)

ただし、 $\max(A, B)$ は $A, B$ のうち大きい方を選択する関数である。

【0045】このとき、式(14)、(15)の条件を満足していればモンゴメリーのべき乗剰余演算はすべてモンゴメリーの剰余乗算の単純な繰り返しによって実現することができる。従って、図3に示すように式(9)～(13)に対してセレクト $S$ によって入力を選択するだけでべき乗剰余演算が実行できる。

【0046】なお、図3の回路で、2つのセレクト $S$ は、選択可能なフィードバック入力として、一方に $C_1$ 、他方に $C_1, M_1$ を一時蓄えるメモリを具えるものとする。このようなメモリは、2つのセレクト $S$ の前に設けて、両セレクト $S$ が共通に利用できるようにしてもよいことはもちろんである。また、このようなセレクト $S$ における入力の切替のためには、例えば、 $e$ をシフトレジスタに記憶させ、 $e_1$ を上位ビットから順次出力させ、その出力を受けて、 $e_1=1$ であるか、および1 > 1であるかの判定を行い、切替信号を出力する制御部 50

(論理回路やカウンタなどにより構成できる)を設ければよい。

【0047】このとき、式(14)、(15)の条件を満足していればモンゴメリーのべき乗剰余演算はすべてモンゴメリーの剰余乗算の単純な繰り返しによって実現することができる。ただし、式(14)、(15)から $u>0$ であるので、演算結果である $C$ だけは、 $C < N$ となるように補正しなければならない。

【0048】従来のイブンの手法では、このような補正をモンゴメリーの剰余乗算を行う度に行わなければならないが、本方式はモンゴメリーのべき乗剰余演算の終了後に1度だけ補正を行えばよい。また、この補正は簡単な処理であるので、以下に示すモンゴメリーのべき乗剰余演算のための回路規模や処理速度に比べてほとんど影響しない。

【0049】〔モンゴメリーのべき乗剰余演算2〕また、べき乗剰余演算： $C = M^e \bmod N$ は次のようにしても実行できる。

【0050】

INPUT  $M, e, N, R_1$

$M_1 = M \cdot R_1 \cdot R^{-1} \bmod N$

$C_1 = 1 \cdot R_1 \cdot R^{-1} \bmod N$

FOR  $i=1$  TO  $t$

IF  $e_i = 1$  THEN  $C_1 = C_1 \cdot M_1 \cdot R^{-1} \bmod N$

IF  $i < t$  THEN  $M_1 = M_1 \cdot M_1 \cdot R^{-1} \bmod N$

NEXT

$C = C_1 \cdot 1 \cdot R^{-1} \bmod N$

この場合も、式(14)、(15)の条件を用いればモンゴメリーの剰余乗算の単純な繰り返しによって $C$ が演算できることは明らかである。また、図3の回路で、2つのセレクト $S$ がそれぞれ $C_1, M_1$ を選択可能とすると共に、2つのセレクト $S$ が、ともに $M_1$ を選択可能とするだけで、同様にべき乗剰余演算が実行できることは明らかである。

【0051】以上によって、べき乗剰余演算及び剰余乗算が式(16)を演算する演算回路のみによって実行できることが示された。

【0052】

$Z = X \cdot Y \cdot R^{-1} \bmod N$  (16)

また、これを式(7)に示すモンゴメリーの剰余乗算を用いて演算する場合には、式(14)、(15)の条件を満足する入力値を用いることによって、モンゴメリーの剰余乗算の単純な繰り返しによって剰余乗算及びべき乗剰余演算が実行できることも示された。

【0053】式(16)または式(7)は整数演算であるので、その演算回路及び方法は種々の手法によって実

現できる。例えば、CPU等を用いれば簡単に実現できることは明らかである。

【0054】従って、式(16)または式(7)を実行する共通の演算回路及び方法によって、剰余乗算及びべき乗剰余演算を用いた種々の暗号システムが効率的に構成できる。

【0055】〔モンゴメリーの剰余乗算及びべき乗剰余乗算〕

$$\begin{aligned} A_k &= A_{k-1} \cdot Y^{k-1} + A_{k-2} \cdot Y^{k-2} + \dots + A_1 \cdot Y + A_0 \\ B_k &= B_{k-1} \cdot X^{k-1} + B_{k-2} \cdot X^{k-2} + \dots + B_1 \cdot X + B_0 \\ N &= N_{k-1} \cdot X^{k-1} + N_{k-2} \cdot X^{k-2} + \dots + N_1 \cdot X + N_0 \\ T_k &= T_{k-1} \cdot X^{k-1} + T_{k-2} \cdot X^{k-2} + \dots + T_1 \cdot X + T_0 \end{aligned} \quad (17)$$

ここで、 $A_i (i=0, \dots, k-1, n+u < i \text{ で } A_i = 0)$  は  $A_k$  を下位桁から  $v$  ビット毎に分割したビット系列を表し、 $B_j, N_j, T_j (j=0, \dots, m-1)$  は各々  $B_k, N, T_k$  について下位桁から  $d$  ビット毎に分割したビット系列を表す。この場合、モンゴメリーの剰余乗算は次の演算を  $i=0$  から  $k$  まで繰り返すことによって求められる。ただし、 $T_1$  は1回目の演算における  $T_k$  の値を意味し、式(16)における  $T_1$  とは異なる。

【0057】

$$T_{i+1} = (T_{i+1} + A_i \cdot B_k \cdot Y + M_{i-1} \cdot N) / Y \quad (18)$$

ただし、 $M_{i-1} = (T_{i-1} \bmod Y) \cdot N_0' \bmod Y, T_{-1} = 0, N_0' = N_0 \bmod Y$

この演算を並列処理で実現するために、 $B_k, N$  を  $B_j, N_j$  を用いて表すと次のようになる。

【0058】アルゴリズム1:

```
FOR i=0 TO k
  Mi-1 = dwv (dwv (Ti-1,0) · N0')
  FOR j=0 TO m-1
    Ri,1 = Ti-1,1 + Li-2,1+1 · X / Y2 + Y · Ai · Bj
    + Mi-1 · Nj
    Li,1 = dwv (Ri,1)
    Ti,1 = (Ri,1 - Li,1) / Y
  NEXT j
NEXT i
```

ただし、 $dw_v(Z) = Z \bmod 2^d$

$up_v(Z) = (Z - dw_v(Z)) / 2^d$

$T_{i,1}, L_{i,1}$  の初期値は全て0

アルゴリズム1において、 $Y \cdot A_i \cdot B_j, L_{i-2,1+1} \cdot X / Y^2$ , および  $T_{i,1} = (dw_{v+1}(R_{i,1}) - L_{i,1}) / Y$  等の定数  $X=2^d, Y=2^v$  による乗除算は他の値に対してビットをずらすことによって実現される。従って、 $T_{i,1}$  に関する演算は  $R_{i,1}$  のLSBに対して  $v$  ビット目から  $d+v-1$  ビット目までの値を  $T_{i,1}$  とすることを意味する。ただし、 $L_{i,1}$  は  $R_{i,1}$  のLSBから  $v-1$  ビット目までの値である。このように  $T_{i,1}$  を得るための  $1/Y$  演算を  $R_{i,1}$  毎の下位へのビットシフトによって実現しているので、 $L_{i-2,1+1}$  は  $R_{i,1}$  を演算するときに用いられ、 $X/Y^2$  によって桁を合わせて演算される。

\*回路の実施例1)  $T_k = A_k \cdot B_k \cdot R^{-1} \bmod N$  ( $A_k, B_k < 2^{u+v}, R=2^{u+v}, N < 2^v$  整数,  $u, v$  は式(14), (15)の条件を満たす)の剰余乗算を考える。 $A_k$  を  $v$  ビット毎、 $B_k, N, T_k$  を  $d$  ビット毎に分割すると、次のように表せる。ただし、 $n+r \leq m \cdot d, n+r \leq k \cdot v, X=2^d, Y=2^v (v \leq d)$ 。

【0056】

【0059】図2はアルゴリズムを行う回路である。

アルゴリズム1において  $i$  はクロックを意味し、 $j$  は図3におけるレジスタ ( $R$ ) の位置に対応し、右から左に  $R_{i,0}$  から  $R_{i,m-1}$  のレジスタを示す。

【0060】以下、簡単のために  $v=1$  の場合について図2の回路と動作を説明する。図2において  $B_j, N_j (j=0, \dots, m-1)$  及び  $N_0'$  は、各々に記された値を乗数として持つ  $d$  ビットの乗算器を示し、 $d$  個のANDによ

て実現できる。 $N$  が奇数であれば、 $N_0' = 1$  であるので、 $M_{i-1}$  を演算する乗算器は省略でき、 $T_{i-1,0}$  のLSBをとして出力する。また、+で示される加算器の入力及び出力は次のようになる。下部の乗算器からの出力  $M_{i-1} \cdot N_j$  は  $d$  ビット、上部の乗算器からの出力  $A_i \cdot B_j$  も  $d$  ビットであるが、その値を2倍するために  $M_{i-1} \cdot N_j$  に対して1ビット上位桁にシフトして入力する。レジスタからの入力  $T_{i-1,1}$  は、 $R_{i-1,1}$  のLSBから2ビット目からを1ビット下位にシフトさせ、 $M_{i-1} \cdot N_j$  と同位の値として入力する。 $L_{i-2,1+1} \cdot 2^{d-2}$  は2つ前のPEからの1ビット出力  $L_{i-2,1+1}$  を  $M_{i-1} \cdot N_j$  のLSBから  $d-1$  ビット目に入力することを意味する。この場合、 $T_{i-1,1} < 2^{d+2}$  であれば、加算器からの出力は  $d+3$  ビットとなる。従って、加算器からの出力を受けるレジスタは各々  $d+3$  ビットレジスタとなる。

【0061】以上のようにして、図4の回路で式(18)の演算が実行でき、 $A_0$  から  $A_k$  まで入力することによってモンゴメリーの剰余乗算が実行できる。

【0062】また、図2は  $v=1$  として説明したが、 $v \leq d$  である  $v$  に対しても同様の手法によってモンゴメリーの剰余乗算を実行できることは明らかである。

【0063】本実施例によるモンゴメリーの剰余乗算回路は非常に小さな回路規模で、高速処理を実現する。

【0064】〔モンゴメリーの剰余乗算及びべき乗剰余回路の実施例2〕この演算をシストリックアレイで実現するために、 $B_k, N$  を  $B_j, N_j$  を用いて表すと次のようになる。

【0065】アルゴリズム2:

```
FOR i=0 TO k
  Mi-1 = dwv (dwv (Ti-1,0) · N0')
  FOR j=0 TO m-1
```

15

$R_{i,j} = T_{i-1,j} + C_{i,j-1} + L_{i-2,j+1} \cdot X / Y^2 + Y \cdot A_{i,j} \cdot B_{i,j} + M_{i-1} \cdot N_j$   
 $L_{i,j} = dw_{i,j}(R_{i,j})$   
 $T_{i,j} = (dw_{i,v}(R_{i,j}) - L_{i,j}) / Y$   
 $C_{i,j} = up_{i,v}(R_{i,j})$   
 NEXT  
 NEXT

ただし、 $dw_i(Z) = Z \bmod 2^d$

$up_i(Z) = (Z - dw_i(Z)) / 2^d$

$T_{i,j}, C_{i,j}, L_{i,j}$  の初期値は全て0

アルゴリズム2において、 $C_{i,j-1}$  は桁上がりとして  $R_{i,j}$  を演算する時に用いられる。また、 $Y \cdot A_{i,j} \cdot B_{i,j}$ 、 $L_{i-2,j+1} \cdot X / Y^2$ 、および  $T_{i,j} = (dw_{i,v}(R_{i,j}) - L_{i,j}) / Y$  等の  $X, Y$  を定数としてもつ演算は他の値に対してビットをずらすことによって実現される。従って、 $T_{i,j}$  に関する演算は  $R_{i,j}$  の LSB に対して  $v$  ビット目から  $d+v-1$  ビット目までの値を  $T_{i,j}$  とすることを意味する。

【0066】ただし、 $L_{i,j}$  は  $R_{i,j}$  の LSB から  $v-1$  ビット目までの値である。このように  $T_{i,j}$  を得るための  $1/Y$  演算を  $R_{i,j}$  毎の下位へのビットシフトによって実現しているため、 $L_{i-2,j+1}$  は  $R_{i,j}$  を演算するときに用いられ、 $X/Y^2$  によって桁を合わせて演算される。

【0067】図5はアルゴリズム2において  $R_{i,j}, L_{i,j}, T_{i,j}, C_{i,j}$  を演算する回路である。図6は図5の回路を1つのPE (プロセッシング・エレメント) として、それを縦列に接続したシストリックアレイである。アルゴリズム2において、 $j$  はクロックを意味し、 $i$  は図6におけるPEの位置に対応し、左から右に  $i = 0$  (#1) から  $i = k$  (# $k+1$ ) のPEを示す。

【0068】図6において# $i+1$  番目のPEは  $A_i$  ( $i = 0, \dots, k$ ) の値が内部レジスタに設定されており、PE間には  $B_{i,j}$  と  $B_{i+1,j}$ 、 $D_{i,j}$  と  $D_{i+1,j}$ 、 $T_{i,j}$  と  $T_{i+1,j}$ 、 $L_{i,j}$  と  $L_{i+1,j}$ 、 $M_{i,j}$  と  $M_{i+1,j}$ 、 $N_{i,j}$  と  $N_{i+1,j}$  が各々接続されている。また、#1のPEの  $B_{i,j}$ 、 $N_{i,j}$  には各々  $B_j$ 、 $N_j$  ( $j = 0, \dots, m-1$ ) が下位桁から順に入力され、 $D_{i,j}$ 、 $T_{i,j}$ 、 $L_{i,j}$ 、 $M_{i,j}$  の入力には各々0が設定されている。

【0069】以下、簡単のために、 $v=1$  の場合について図5の回路と動作を説明する。図5において  $\times$  は乗算器を示し、 $d$  個のアンドによって実現できる。 $R_1 \sim R_3$  は各々  $A_i$ 、 $M_{i-1}$ 、 $N_0'$  を保持する1ビットレジスタである。 $N$  が奇数であれば、 $N_0' = 1$  であるのを演算する乗算器と  $N_0'$  を保持する  $R_3$  は省略され、 $R_2$  は  $T_{i-1,0}$  の LSB をとして保持する。また、 $R_4$ 、 $R_5$  は  $B_{i,j}$ 、 $N_{i,j}$  からの入力を1クロック遅らせて次のPEに送るための  $d$  ビットレジスタである。 $+$  で示される加算器の入力及び出力は次のようになる。下部の乗算器からの出力  $M_{i-1} \cdot N_j$  は  $d$  ビット、上部の乗算器からの出力  $A_i \cdot B_j$  も  $d$  ビットであるが、その値を2倍するため

16

に  $M_{i-1} \cdot N_j$  に対して1ビット上位桁にシフトして入力する。前PEからの入力  $T_{i-1,j}$  は、 $R_{i-1,j}$  の LSB から2ビット目から  $d+1$  ビット目までの  $d$  ビットを1ビット下位にシフトさせ、 $M_{i-1} \cdot N_j$  と同位の値として入力する。 $L_{i-2,j+1} \cdot 2^{d-2}$  は2つ前のPEからの1ビット出力  $L_{i-2,j+1}$  を  $M_{i-1} \cdot N_j$  の LSB から  $d-1$  ビット目に入力することを意味する。この場合、桁上がりビットである  $C_{i,j-1}$  は  $C_{i,j-1} < 2^{d+2}$  であれば、加算器からの出力は  $d+3$  ビットとなるので、 $C_{i,j-1}$  は2ビットの値となる。従って、加算器からの出力を受ける  $R_6$  は  $d+3$  ビットレジスタとなる。

【0070】以上により、図5のPE1つで式(18)の演算を実行できることがわかる。このPEを図6のように  $k+1$  個パイプライン状に接続し、クロックに同期させて動作させることによってモンゴメリーの剰余乗算が高速に実行できる。

【0071】ただし、図6のアレイからの最終出力は  $k+1$  番目のPEからの出力  $T_{k,j}$ 、 $L_{k,j}$  と、 $k$  番目のPEからの出力  $L_{k-1,j+1}$  に分離されているので、アルゴリズム2の処理の後、次のような演算を行う。

【0072】アルゴリズム3:

FOR  $j = 0$  TO  $m-1$

$R_{k+1,j} = T_{k,j} + C_{k+1,j-1} + L_{k-1,j+1} \cdot X / Y^2$

$T_{k+1,j} = dw_{k+1,j}(R_{k+1,j})$

$C_{k+1,j} = up_{k+1,j}(R_{k+1,j})$

$T_{k+2,j} = T_{k+1,j} + C_{k+2,j-1} + L_{k,j+1} \cdot X / Y$

$C_{k+2,j} = up_{k+2,j}(T_{k+2,j})$

NEXT

ここで、 $T_{k+2,j}$  が  $T_k$  を分割したビット系列  $T_i$  ( $i = 0, \dots, m-1$ ) となる。 $R_{k+1,j}$  の演算はアルゴリズム2において  $A_i = M_{i-1} = 0$  とした場合の演算と同様であるので、図5のPEによって実現される。 $T_{k+2,j}$  の演算は、ほぼ  $R_{k-1,j}$  の演算と同様であるが、 $T_{k+1,j}$ 、 $C_{k+1,j}$  は  $1/2$  演算が行われず  $L_{k,j+1}$  も  $L_{k-1,j+1}$  に対して1ビット上位桁で加算される。従って、図7に示すように図5のPEの加算器とレジスタ  $R_6$  の LSB の下に1ビット分のハーフアダー (HA) とレジスタ ( $R_7$ ) を用意し、HAには前PEの出力  $R_{i-1,j}$  の LSB と、 $C_{k+2,j-1}$  を入力し (この時、 $C_{k+2,j-1}$  は高々1ビットの値である)、その加算結果を新たに用意したレジスタへ、キャリービットを加算器へのキャリーとして入力する。このようにすれば  $L_{k,j+1}$  は自動的に1ビット上位の桁として加算される。従って、 $T_{k+2,j}$  を演算するPEは他のPEと異なるが、 $i = k+2$  の時のみ加算器へのキャリーとしてハーフアダーからのキャリーを選択する1ビットのセレクトを加えれば全てのPEを1つのPEで実現できる。

【0073】従って、 $T_k$  を得るには図6のPEとして図7に示したPEを用い、さらに図6のアレイの後にPEを2個追加した計  $k+3$  個のPEを用いたアレイによ

17

ってモンゴメリーの剰余乗算が高速演算される。

【0074】また、図5及び図7は $v=1$ として説明したが、 $v \leq d$ である $v$ に対しても同様の手法によってモンゴメリーの剰余乗算を実行できることは明かである。

【0075】〔モンゴメリーの剰余乗算及びべき乗剰余回路の実施例3〕また、次のようなシストリックアレイを構成することができる。式(17)の演算を次のようなアルゴリズムによって実行する。

【0076】アルゴリズム4:

```
FOR i=0 TO k
  Mi-1 = dw, (dw, (Ti-1,1) · N0')
  FOR j=0 TO m
    Ri,1 = Ti-1,1+1 + Ci,1-1 + Ai · Bj-1 + Mi-1 · Nj
    Ti,1 = dw, (Ri,1)
    Ci,1 = up, (Ri,1)
  NEXT
NEXT
```

アルゴリズム4における $R_{i,1}$ 演算のアルゴリズム2との違いは、 $M_{i-1} \cdot N_j$ に対する $A_i \cdot B_j$ 及び $T_{i-1,1}$ の桁の違いをビットずれではなく、用いる $B_j$ 及び $T_{i,1}$ のjに関する係数をずらすことによって実現している点である。従って、アルゴリズム2でビットずれのために生じる値 $L_{i,1}$ はアルゴリズム4では生じない。

【0077】アルゴリズム4の $R_{i,1}$ 、 $T_{i,1}$ 、 $C_{i,1}$ の演算を実行するPE及びシストリックアレイを図8、9に示す。アルゴリズム4のj、iもそれぞれ、クロック及びPEの位置に対応する。また、図9においても#1+1のPEには $A_i$  ( $i=0, \dots, k$ )の値が内部レジスタに設定されており、PE間には $B_{i,1}$ と $B_{0,1}$ 、 $T_{i,1}$ と $T_{0,1}$ 、 $M_{i,1}$ と $M_{0,1}$ 、 $N_{i,1}$ と $N_{0,1}$ が各々接続されている。また、#1のPEの $T_{i,1}$ 、 $M_{i,1}$ の入力には各々0が設定されているが、 $B_{i,1}$ 、 $N_{i,1}$ には各々 $B_j$ 、 $N_j$  ( $j=0, \dots, m-1$ )が下位桁から順に入力される。ただし、アルゴリズム1の場合と異なり $B_j$ は $N_j$ に対して1クロック遅れで入力される。

【0078】以降、簡単のために $v=d$ の場合について説明する。図8において、 $\times$ はdビット・dビットの乗算器を示し、 $+$ は加算器を示す。加算器の入力及び出力は次のようになる。上部の乗算器からの出力 $A_i \cdot B_{j-1}$ と下部の乗算器からの出力 $M_{i-1} \cdot N_j$ は各々2・dビットであり、前PEからの出力 $T_{i-1,1+1}$ はdビットの値である。従って、 $C_{i,1-1}$ が $C_{i,1-1} < 2^{2 \cdot d+1}$ であれば加算器からの出力は2・d+2ビットの値である。また、 $R_1 \sim R_7$ はdビットのレジスタであり、加算器からの出力を受ける $R_8$ は2・d+2ビットのレジスタである。レジスタ $R_8$ は、LSBからdビット目までを $T_{i-1,1+1}$ として次のPEへ出力し、上位のd+2ビットを $C_{i,1-1}$ として加算器へフィードバックする。

【0079】図8では $N_j$ は $B_j$ に比べて1クロック前

18

に入力されるので $A_i \cdot B_{j-1}$ と $M_{i-1} \cdot N_j$ が同時に演算される。また、 $T_{i-1,1+1}$ を $A_i \cdot B_{j-1}$ 及び $M_{i-1} \cdot N_j$ と同時に演算するために、 $B_{i,1}$ 、 $N_{i,1}$ から入力される $B_j$ 、 $N_j$ は2クロック遅らされて次のPEに出力される。

【0080】従って、図8のPEによっても式(17)の演算を行うことができ、図9のシストリックアレイによってモンゴメリーの剰余乗算が高速に実現できることがわかる。この場合、アルゴリズムに相当する処理は必要なく、図9に示すようにm+1個のPEによってアレイが構成できる。

【0081】また、図8は $v=d$ として説明したが、 $v < d$ である $v$ に対しても同様の手法によってモンゴメリーの剰余乗算を実行できることは明らかである。

【0082】〔モンゴメリーの剰余乗算及びべき乗剰余回路の実施例4〕また、実施例3に示したアルゴリズム4を実行する場合、 $A_i$ は予めPEに設定している必要はなく、図10のように $A_{i,1}$ から $A_i$  ( $i=0, \dots, k-1$ )を下位桁から $N_j$ に同期させて入力させ、図11のように $A_{i,1}$ と $A_{0,1}$ を接続して次のPEに送る構成にしてもよい。この場合、 $A_i$  ( $i=0, \dots, k-1$ )は $B_j$  ( $j=0, \dots, m-1$ )よりも1クロック先に入力されるので、#1のPEにおいて $A_0$ が入力されると同時に $R_1$ のレジスタに $A_0$ を保持すると、 $A_0 \cdot B_j$  ( $j=0, \dots, m-1$ )を演算を全てのjに対して実行することができる。また、 $B_j$ は2クロック遅れて次のPEに入力されるが、 $A_i$ は1クロックしか遅れないので、#1-1のPEにおいて $A_i$ が $B_j$  ( $j=0, \dots, m-1$ )の前に入力され保持できたとすると、#1のPEでは $A_{i+1}$ が $B_j$  ( $j=0, \dots, m-1$ )の前に入力され保持できる。従って、#1のPEにおいて無理なく $A_{i+1} \cdot B_j$  ( $j=0, \dots, m-1$ )の演算が実行できる。従って、回路規模及び処理速度を変えことなくアルゴリズム4を図10、11のPE及びシストリックアレイによっても実現できる。

【0083】〔モンゴメリーの剰余乗算及びべき乗剰余回路の実施例5〕モンゴメリーのべき乗剰余演算は式(17)の演算の繰り返しによって実行できる。図10及び図8、10に示したPEは式(17)の演算を行うことができるので、図12のようにメモリと組み合わせれば、1つのPEを $(3 \cdot t / 2 + 2) \cdot q$ 回 ( $q$ はモンゴメリーの剰余乗算アレイを構成するために必要なPEの数で、図7のPEでは $k+3$ 個、図8、10のでは $k-2$ 個)用いてモンゴメリーのべき乗剰余演算を処理できる。もし、p個のPEを用いるならば、 $(3 \cdot t / 2 + 2) \cdot q / p$ 回の繰り返しでモンゴメリーのべき乗剰余演算を処理できる。処理速度は繰り返し回数に反比例するので、この方式は処理速度をPEの数に比例させることができ、また、PEの数による処理速度の高速化または回路規模の小型化に対して同じ効率のべき乗剰余演算回路を構成することができる。

【0084】従って、図13に示すような装置化が可能になる。図13において、SYMC (Systolic Modular Exponentiation Chip) と表されているのは、 $p$ 個のPEを縦列接続したものをチップ化したものである。 $p$ は $1 \leq p \leq (3 \cdot t / 2 + 2) \cdot q$ であれば任意であるので、任意の回路規模のチップを構成することができる。また、SYMCは回路構成に規則性をもつので装置化及びチップ化が非常にしやすい。また、処理速度はSYMCの数に比例して高速化できるので、図13に示すようにSYMCを従属に接続するだけでよい。この場合、SYMCの処理回数を変える必要があるが、これは制御回路を外部からプログラミング可能なROM等によって構成することによって容易に実現できる。

【0085】(モンゴメリーの剰余乗算及びべき乗剰余演算回路のその他の実施例) 上記実施例によるアルゴリズムでは1つのPEで行う処理は簡単な整数演算であるので、PEを別にチップ化しなくても通常のDSPやCPU等によってもモンゴメリーのべき乗剰余演算を簡単に実現することができる。

【0086】また、上記実施例はシストリックアレイを基本としているので、回路構成が規則的であり、制御や遅延も局所的であるのでVLSIによる実用化にも最適である。

【0087】また、上記実施例のPEを従属に接続せず独立した演算素子として用い、よく知られたマイクロプログラミング的な手法によって制御して剰余乗算及びべき乗剰余演算を実現することも容易である。

【0088】また、図5、7及び図8、10のPEは式(18)の演算を一括して実行しているが、式(18)を種々に分解した演算素子によって最終的に式(18)の演算を実行する場合も本発明は含んでいる。

【0089】また、本発明をシストリックアレイによって実行する場合、制御信号もデータと同時に入力させることができ、制御信号の伝送レジスタまで含んだものをPEとすることもできる。

【0090】以上によってシストリックアレイを用いたべき乗剰余演算及び剰余乗算回路及び方法の構成法が示された。この方式はイブンによる手法の欠点をすべて解決した回路及び方法を提供する。それによって、次のような効果を持つ効率的な暗号システムを構成することができる。

【0091】高速な暗号システムが必要な場合、本発明によるべき乗剰余演算及び剰余乗算回路をVLSI等によって構成すればよい。この場合、本発明によるべき乗剰余演算及び剰余乗算回路は簡単なPEによる規則構造を持ち、かつ、PEの制御とPE内の遅延時間は局所的であるので、VLSIに最適である。これによって、高速な暗号システムが構成される。

【0092】また、高速性よりも小型回路による暗号システムが要求される場合は、本発明によるべき乗剰余演

算及び剰余乗算回路をPE数個で構成すればよい。この場合も、PEによる規則構造と制御と遅延時間の局所性といった特徴は失われず回路化しやすい。また、PE内で行われる演算は簡単な整数演算であるので、本発明による演算手順はCPUやDSP等のソフト的な手法によっても簡単な暗号システムを実現することができる。

【0093】また、いくつかのPEからなる小型回路(SYMC等)による暗号装置を構成した後で、高速性が必要となっても図10のように、その小型回路を縦続に接続して行けば、回路規模に比例した高速化が実現できる。従って、暗号装置を作り直すことなく、継ぎ足して行くだけで必要に応じた高速化が簡単にできる暗号システムを実現できる。

【0094】また、1度暗号システムを構成した後で、暗号システムの強度を増すために演算する整数のビット数を増す場合も、同一の回路または、PEの数を増した同様の回路によって対応することができる。これは、本発明のべき乗剰余演算及び剰余乗算回路が回路規模と処理回数を簡単にトレードオフできるので、演算すべき整数のビット数の違いを処理回数の違いに帰着できるためである。従って、システムの暗号的な強度を増す場合にも、暗号装置をつくり直す必要がない。また、演算する整数のビット数を減少させる場合にも、暗号装置をつくり直すことのない暗号システムを実現することができる。

【0095】以上のような効果は、特願平3-225986号でも述べているように従来のべき乗剰余演算及び剰余乗算による暗号装置では実現できないものである。従って、本発明によるべき乗剰余演算及び剰余乗算回路及び方法を用いることによって柔軟で拡張性のある暗号システムを構成することができる。

【0096】次に、特願平3-225986号のシストリックアレイ(以後、アレイ0と呼ぶ)と本発明の実施例1に示したシストリックアレイ(アレイ1)、実施例2、3に示したシストリックアレイ(アレイ2)の比較を行う。

【0097】アレイ0は除算を伴うために剰余テーブル(ROM等)によって剰余演算を行うが、アレイ1、2はすべて乗算によって剰余乗算が演算できるため1クロックに必要な処理時間がアレイ0に比べて短いため高速処理できる。また、アレイ1、2はROM等の剰余テーブルを用いず、ANDゲートのような回路規模のくすることができる。また、アレイ0はキャリービットの処理のためのPEを必要とするが、アレイ1、2は必要としないので必要なPEの数がアレイ0に比べて少ない。従って、同じ程度の回路規模を用いた場合、アレイ1、2はアレイ0に対して数倍の高速処理が行える。

【0098】また、アレイ1はアレイ0に比べてPEの種類が少なく図7に示したようにPEを共通化しやすい。また、アレイ2は1種類のPEのみで構成できる。従って、アレイ1、2はアレイ0よりも容易に回路化す

ることができ、アレイ0より無駄のない回路を構成することができる。

【0099】また、アレイ0が剰余テーブルを用いている場合、演算すべき各整数のビット数の増加に対してアレイ1, 2はアレイ0よりも柔軟性が高い。これは、アレイ0がROM等の剰余テーブルを用いている場合、テーブルの最大容量によって各整数のビット数が制限されるためである。これに対し、アレイ1, 2は乗算によって剰余が演算されるので剰余テーブルが必要なく演算する整数のビット数に制限がない。ただし、演算する整数のビット数が減少する場合（中国人の剰余定理を用いる場合等）には、アレイ0～アレイ2の柔軟性は同じであり、制限がない。よって、演算する整数のビット数の変化に対して、アレイ1, 2はアレイ0と異なり制限がない。従って、アレイ1, 2は演算する整数のビット数の異なる演算に対してもSYMC等の回路を全く作り直す必要がない。

【0100】以上のことから、上記実施例による剰余乗算及びべき乗剰余方式を用いた暗号装置は、上述の効果を最も小さな回路で高速に、かつ柔軟に実現できる暗号システムを提供することができる。

【0101】〔モンゴメリーの剰余乗算及びべき乗剰余回路の実施例6〕1回目の演算における $T_1$ の値 $T_{1,1}$ を、式(18)における $T_1$ とは異なり、

$$T_{1,1} = (T_{1,1-1} / Y + A_1 \cdot B_1) + M_1 \cdot N \quad (19)$$

ただし、 $M_1 = ((T_{1,1-1} / Y + A_1 \cdot B_1) \bmod Y) \cdot N_0' \bmod Y$ ,

$$T_{1,1} = 0, N_0' = N' \bmod Y$$

この演算を複数のPEによる並列処理で実現するために、 $B_1$ ,  $N$ を $B_1$ ,  $N_1$ に分解して次のように表す。

アルゴリズム5:

```
FOR i=0 TO k-1
FOR j=0 TO m-1
 $S_{i,j} = T_{i-1,j} / Y + A_i \cdot B_j + C_{i,j-1}$ 
 $M_i = dw_i(dw_i(S_{i,0}) \cdot N_0')$ 
 $R_{i,j} = S_{i,j} + M_i \cdot N_j + L_{i-1,j+1} \cdot X$ 
 $L_{i,j} = dw_i(R_{i,j})$ 
 $T_{i,j} = dw_{d+v}(R_{i,j}) - L_{i,j}$ 
 $C_{i,j} = up_{d+v}(R_{i,j})$ 
NEXT
NEXT
```

ただし、 $dw_i(Z) = Z \bmod 2^d$

$$up_d(Z) = (Z - dw_d(Z)) / 2^d$$

$T_{1,1}$ ,  $C_{1,1}$ ,  $L_{1,1}$ の初期値は全て0

アルゴリズム5において、 $C_{i,j-1}$ は桁上がりとして $S_{i,j}$ を演算する時に用いられる。また、 $L_{i-1,j+1} \cdot X$ , および $T_{i-1,j} / Y$ 等の $X$ ,  $Y$ を定数としてもつ演算は他の値に対してビットをずらすことによって実現できる。従って、 $T_{i,j}$ に関する演算は $R_{i,j}$ のLSBに対して $v$ ビット目から $d+v-1$ ビット目までの値を $T$

$_{i,j}$ とすることを意味する。ただし、 $L_{i,j}$ は $R_{i,j}$ のLSBから $v-1$ ビット目までの値である。このように $T_{i,j}$ を得るための $1/Y$ 演算を $R_{i,j}$ 毎の下位へのビットシフトによって実現しているため、 $L_{i-1,j+1}$ は $R_{i,j}$ 演算するときに用いられ、 $X$ によって桁を合わせて演算される。

【0102】アルゴリズム5において1を処理回数、 $j$ をクロックと考えると各 $j$ に対して演算しなければならないのは $S_{i,j}$ 及び $R_{i,j}$ のみであり( $L_{i,j}$ ,  $T_{i,j}$ ,  $C_{i,j}$ はビットシフトのみで実現される)、 $S_{i,j}$ 及び $R_{i,j}$ は次の同型の演算によって実現される。

【0103】

$$f = d / y + a \cdot b + c \cdot x \quad (20)$$

ただし、 $y$ は $2^v$ または1、 $x$ は $2^d$ または1

$x$ ,  $y$ はビットシフトの有無であるので、式(20)は図14に示すPEによって演算できる。

【0104】以下、簡単のために $v=1$ の場合について図14の回路と動作を説明する。図14において $\times$ は乗算器を示し、 $d$ 個のアンドによって実現できる。 $R1$ は $a$ を保持する1ビットレジスタである。 $S1$ ,  $S2$ は入力 $d$ ,  $c$ を $y$ または $x$ によってビットシフトさせるかどうかを選択するセレクタである。 $+$ で示される加算器は乗算器からの出力 $a \cdot b$ とセレクタからの出力 $d/y$ 及び $c \cdot x$ の加算を行い $f$ を演算する。 $R2$ は加算器からの出力を保持するレジスタである。以上によって、図14のPEで式(20)が演算できることが分かる。

【0105】従って、アルゴリズム5の各演算は図15のように図14のPEを2つ組み合わせれば求めることができる。ただし、図15の $B_{1,1}$ ,  $N_{1,1}$ には各々 $B_1$ ,  $N_1(j=0, \dots, m-1)$ が下位桁から順に入力されるとする。この場合、左のPEで $S_{i,j}$ が演算され、右のPEで $R_{i,j}$ が演算される。また、 $v=1$ であるので $N$ が奇数であれば、 $N_0'=1$ となり、 $S_{i,0}$ のLSBが $M_i$ となり、右のPEのレジスタ $R1$ に保持される。また、 $R_{i,j}$ は $C_{i,j}$ ,  $T_{i,j}$ ,  $L_{i,j}$ に分割されて表示されている。よって、図15を $k$ 個または図14のPEを $2 \cdot k$ 個用いればアルゴリズム5が実行できることは明らかである。以上から、図14のPEによって式(19)を効率的に並列処理できることがわかる。

【0106】また、図14, 15は $v=1$ として説明したが、 $v < d$ である $v$ に対しても同様の手法によってモンゴメリーの剰余乗算を実行できることは明らかである。

【0107】〔モンゴメリーの剰余乗算及びべき乗剰余回路の実施例7〕また、次のようなシストリックアレイを構成することができる。式(14)の演算を次のようなアルゴリズムによって実行する。

アルゴリズム6:

```
FOR i=0 TO k
FOR j=0 TO m
```

23

$S_{i,j} = T_{i-1,j+1} + up_r (S_{i,j-1}) + A_i \cdot B_j$   
 $M_i = dw_r (dw_r (S_{i,0}) \cdot N_0')$   
 $R_{i,j} = dw_r (S_{i,j}) + up_r (R_{i,j-1}) + M_i \cdot N_j$   
 $T_{i,j} = dw_r (R_{i,j})$

NEXT

NEXT

アルゴリズム6のアルゴリズム5に対する違いは、式  
 (19)の $T_{i-1,j+1}$ をビットずれではなく、クロック  
 のずれによって実現している点である。従って、アルゴ  
 リズム5でビットずれのために生じる値 $L_{i,j}$ は、アル  
 ゴリズム6では生じない。

【0108】アルゴリズム6を実行するPE及びシスト  
 リックアレイを図16、17に示す。アルゴリズム6の  
 $j$ 、 $i$ もまた、それぞれがクロック及び処理回数に対応  
 する。また、図17の $B_{i,j}$ 、 $N_{i,j}$ には各々 $B_i$ 、 $N_i$  ( $i=$   
 $0, \dots, m-1$ ) が下位桁から順に入力される。

【0109】以降、簡単のために $v=d$ の場合について  
 説明をする。まず、図16において、 $\times$ は $d$ ビット・ $d$   
 ビットの乗算器を示し、 $+$ は加算器を示す。 $R_1$ は $A_i$   
 または $M_i$ を保持するレジスタであり、 $R_2$ は加算器から  
 の出力を保持するレジスタであり、その $v$ ビット目以  
 上は桁上がりとして1クロック遅れで加算器にフィード  
 バック入力されている。これによって、図17では左の  
 PEにおいて $S_{i,j}$ が演算され、右のPEにおいて $R_{i,j}$   
 が演算されることがわかる。このとき、 $M_i$ は外部  
 にある乗算器によって $N_0'$ と乗算され右のPEへ出力さ  
 れる。従って、図7のPEはアルゴリズム6を並列処理  
 によって実現するための効率的な基本演算子になってい  
 ることがわかる。

【0110】また、図16、17は $v=d$ として説明し  
 たが、 $v < d$ である $v$ に対しても同様の手法によってモ  
 ンゴメリーの剰余乗算を実行できることは明らかであ  
 る。

【0111】〔モンゴメリーの剰余乗算及びべき乗剰余  
 回路の実施例8〕モンゴメリーのべき乗剰余演算は式  
 (19)の演算の繰り返しによって実行できる。図14  
 及び図16に示したPEは式(19)の演算を行うこと  
 ができるので、図18のようにメモリと組み合わせれ  
 ば、1つのPEを $(3 \cdot t/2 + 2) \cdot q$ 回 ( $q$ はモンゴ  
 メリーの剰余乗算アレイを構成するために必要なPEの  
 数で、図14のPEでは $2 \cdot k$ 個、図16のPEでは $2$   
 $\cdot (k+1)$ 個) 用いてモンゴメリーのべき乗剰余演算を  
 処理できる。もし、 $p$ 個のPEを用いるならば、 $(3 \cdot$   
 $t/2 + 2) \cdot q/p$ 回の繰り返しでモンゴメリーのべき  
 乗剰余演算を処理できる。処理速度は繰り返し回数に反  
 比例するので、この方式は処理速度をPEの数に比例さ  
 せることができ、また、PEの数による処理速度の高速  
 化または回路規模の小型化に対して同じ効率のべき乗  
 剰余演算回路を構成することができる。

【0112】従って、図19に示すような装置化が可能

24

になる。図19においてMEC (Modular Exponential  
 on Chip) と表されているのは、 $p$ 個のPEを組み合わ  
 せてチップ化したものである。 $p$ は $1 \leq p \leq (3 \cdot t/$   
 $2 + 2) \cdot q$ であれば任意であるので、任意の回路規模の  
 チップを構成することができる。また、MECは回路構  
 成に規則性をもつので装置化及びチップ化が非常に行い  
 やすい。また、処理速度はMECの数に比例して高速化  
 できるので、図7に示すようにMECを複数用いればよい。  
 この場合、各MECの制御をチップ数に応じて変え  
 る必要があるが、これは制御回路を外部からプログラミ  
 ング可能なROM等によって構成することによって容易  
 に実現できる。

【0113】〔モンゴメリーの剰余乗算及びべき乗剰余  
 演算回路のその他の実施例〕本発明によるアルゴリズム  
 では1つのPEで行う処理は簡単な整数演算であるの  
 で、PEを別にチップ化しなくても通常のDSPやCPU  
 等によってもモンゴメリーのべき乗剰余演算を簡単に  
 実現することができる。

【0114】また、本発明は回路構成が規則的であり、  
 制御や遅延も局所的であるのでVLSIによる実用化に  
 も最適である。

【0115】また、図14、16のPEを組み合わせた  
 ものを1つのPEとして構成することも可能である。

【0116】以上によってPEを用いたべき乗剰余演算  
 及び剰余乗算回路及び方法の構成法が示された。それ  
 によって、次のような効果を持つ効率的な暗号システム  
 を構成することができる。

【0117】高速な暗号システムが必要な場合、本発明  
 によるべき乗剰余演算及び剰余乗算回路をVLSI等  
 によって構成すればよい。この場合、本発明によるべき  
 乗剰余演算及び剰余乗算回路は簡単なPEによる規則構  
 造を持つので、VLSIに最適である。これによって、高  
 速な暗号システムが構成される。

【0118】また、高速性よりも小型回路による暗号シ  
 ステムが要求される場合は、本発明によるべき乗剰余演  
 算及び剰余乗算回路をPE数個で構成すればよい。この  
 場合も、PEによる規則構造といった特徴は失われず回  
 路化しやすい。また、PE内で行われる演算は簡単な整  
 数演算であるので、本発明による演算手順はCPUやD  
 SP等のソフト的な手法によっても簡単な暗号システム  
 を実現することができる。

【0119】また、いくつかのPEからなる小型回路  
 (MEC等) による暗号装置を構成した後で、高速性が  
 必要となっても図19のように、その小型回路を複数用  
 いれば回路規模に比例した高速化が実現できる。従っ  
 て、暗号装置を作り直すことなく、回路を増して行くだ  
 けで必要に応じた高速化が簡単に行える暗号システムを  
 実現できる。

【0120】また、1度暗号システムを構成した後で、  
 暗号システムの強度を増すために演算する整数のビット

数を増す場合も、同一の回路または、P Eの数を増した同様の回路によって対応することができる。これは、本発明のべき乗剰余演算及び剰余乗算回路が回路規模と処理回数を簡単にトレードオフできるので、演算すべき整数のビット数の違いを処理回数の違いに帰着できるためである。従って、システムの暗号的な強度を増す場合にも、暗号装置をつくり直す必要がない。また、演算する整数のビット数を減少させる場合にも、暗号装置をつくり直すことのない暗号システムを実現することができる。

【0121】以上のような効果は、効率的に並列処理を用いない従来のべき乗剰余演算及び剰余乗算による暗号装置では実現できないものである。従って、本発明によるべき乗剰余演算及び剰余乗算回路及び方法を用いることによって柔軟で拡張性のある暗号システムを構成することができる。

【0122】

【発明の効果】以上説明したように、本発明によれば、べき乗剰余演算及び剰余乗算が、

$$Z = X \cdot Y \cdot R^{-1} \bmod N \quad (16)$$

の繰り返し演算によって実行できる。従って、必要な演算は、同一または同型の演算回路によって実行できる。

【0123】また、これをモンゴメリーの剰余乗算

$$Z = X \cdot Y \cdot R^{-1} \bmod N$$

$$= (X \cdot Y + S \cdot N) / R$$

$$\text{ただし、} S = X \cdot Y \cdot N' \bmod N$$

を用いて演算する場合には、条件を満足する入力値を用いることによって、モンゴメリーの剰余乗算の単純な繰り返しによって剰余乗算及びべき乗剰余演算が実行できる。

【0124】また、必要な演算は整数演算であるので、その演算回路が簡単に実現できるようになった。

【0125】従って、共通の演算回路及び方法によって、剰余乗算及びべき乗剰余演算を用いた種々の暗号システムが効率的に構成できるようになった。

【0126】本発明によるモンゴメリーの剰余乗算回路は非常に小さな回路規模で、高速処理を実現する。

【0127】本発明によるシストリックアレイを用いたべき乗剰余演算及び剰余乗算回路によって、次のような効果を持つ効率的な暗号システムを構成することができる。

【0128】高速な暗号システムが必要な場合、本発明によるべき乗剰余演算及び剰余乗算回路をVLSI等によって構成すればよい。この場合、本発明によるべき乗剰余演算及び剰余乗算回路は簡単なP Eによる規則構造を持ち、かつ、P Eの制御とP E内の遅延時間は局所的であるので、VLSIに最適である。これによって、高速な暗号システムが構成される。

【0129】また、高速性よりも回路規模の小型化が暗号システムに要求される場合は、本発明によるべき乗剰

余演算及び剰余乗算回路をP E数個で構成すればよい。この場合も、P Eによる規則構造と制御と遅延時間の局所性といった特徴は失われず回路化しやすい。また、P E内で行われる演算は簡単な整数演算であるので、本発明による演算手順はCPUやDSP等のソフト的な手法によっても簡単な暗号システムを実現することができる。

【0130】また、いくつかのP Eからなる小型回路(SYMC等)による暗号装置を構成した後、高速性が必要となっても、その小型回路を縦続に接続して行くことで、回路規模に比例した高速化が実現できる。従って、暗号装置をまったく新たに作り直すことなく、継ぎ足して行くだけで必要に応じた高速化が簡単に行える暗号システムを実現できる。

【0131】また、回路規模と処理回数を簡単にトレードオフできるので、1度暗号システムを構成した後、暗号システムの強度(解読に対する安全性)を高めるために演算する整数のビット数を増加させる場合も、演算すべき整数のビット数の違いを処理回数の違いに帰着できるため、同一の回路または、P Eの数を増した同様の回路によって対応することができる。従って、この場合、暗号装置を新たに作り直す必要がない。また、演算する整数のビット数を減少させる場合にも、暗号装置をあらためて作り直すことなく対処することができる。

【0132】従って、本発明によるべき乗剰余演算及び剰余乗算回路及び方法を用いることによって柔軟で拡張性のある暗号システムを構成することができる。

【図面の簡単な説明】

【図1】暗号システムを用いる通信系の構成例を示す図である。

【図2】本発明による剰余乗算回路の例を示す図である。

【図3】本発明によるべき乗剰余演算回路の例を示す図である。

【図4】実施例の剰余乗算回路のブロック構成図である。

【図5】実施例のP E(プロセッシング・エレメント)を示す図である。

【図6】図5のP Eを用いたシストリックアレイを示す図である。

【図7】共通化P Eを示す図である。

【図8】他の実施例のP Eを示す図である。

【図9】図8のP Eを用いたシストリックアレイを示す図である。

【図10】他の実施例のP Eを示す図である。

【図11】図10のP Eを用いたシストリックアレイを示す図である。

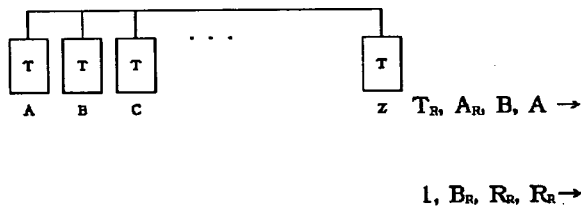
【図12】P Eとメモリを組み合わせた回路

【図13】SYMCを用いたべき乗剰余演算及び剰余乗算回路を示す図である。

- 【図14】実施例2のPEを示す図である。  
 【図15】図14のPEを用いた回路を示す図である。  
 【図16】実施例2のPEを示す図である。  
 【図17】図16のPEを用いた回路を示す図である。  
 【図18】図17の回路とメモリを組み合わせた回路を示す図である。  
 【図19】MECを用いたべき乗剰余演算及び剰余乗算回路を示す図である。  
 【符号の説明】  
 T 通信端末

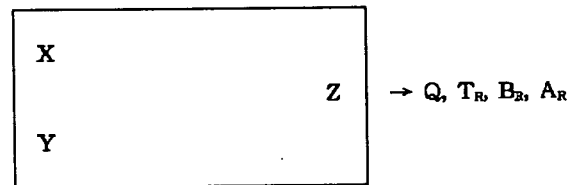
- S, SL セレクタ  
 R, R1 レジスタ  
 H ハーフアダプター  
 + 加算器  
 B<sub>i</sub> 乗算器  
 N<sub>i</sub> 乗算器  
 PE プロセッシング・エレメント  
 MEC べき乗剰余演算チップ  
 SYMC シストリックべき乗剰余演算チップ

【図1】

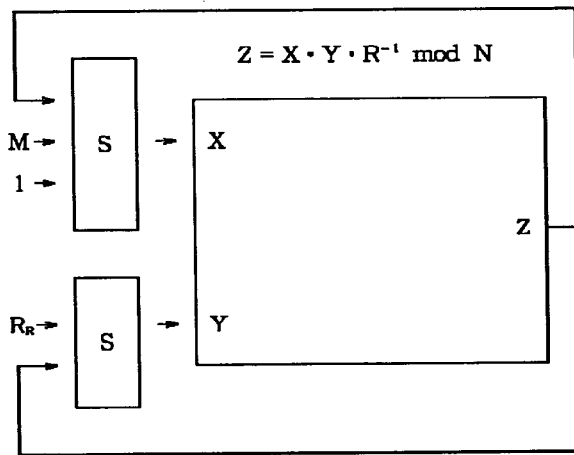


【図2】

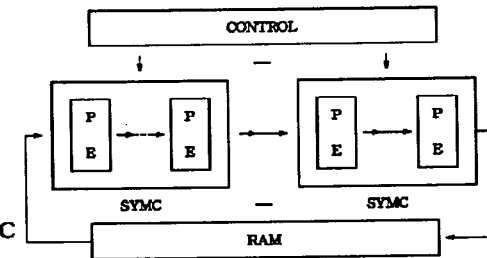
$$Z = X \cdot Y \cdot R^{-1} \bmod N$$



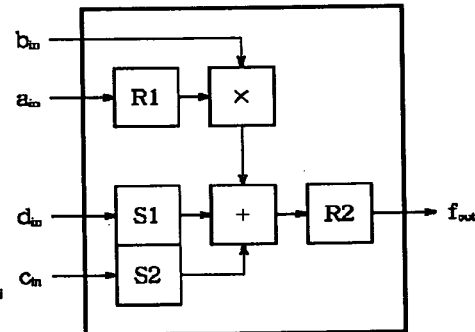
【図3】



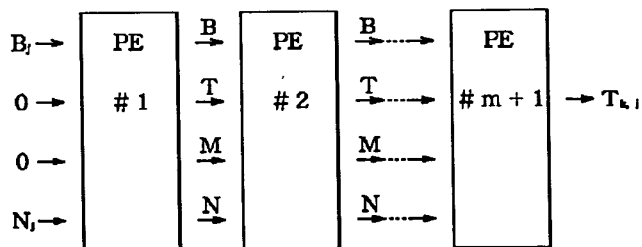
【図13】



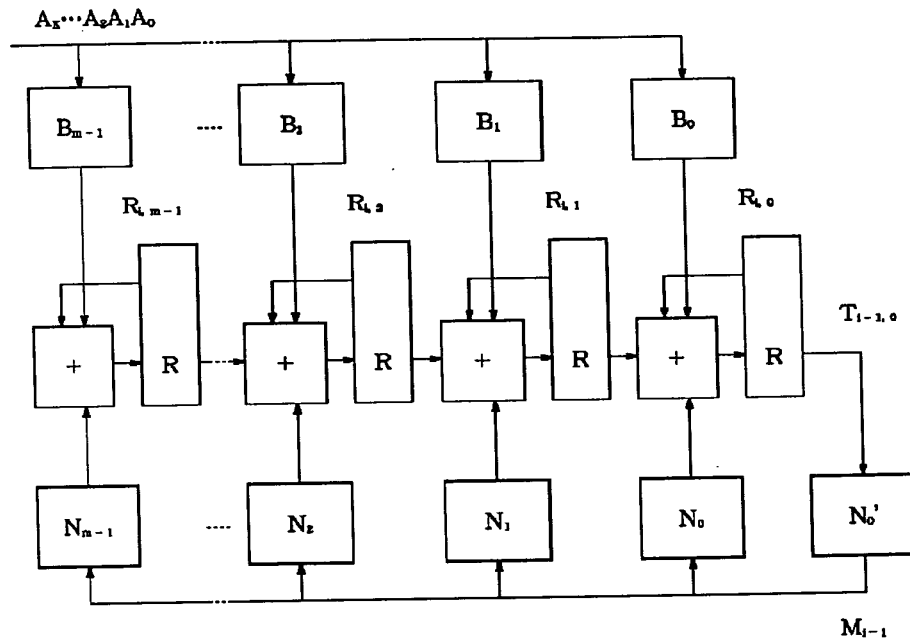
【図14】



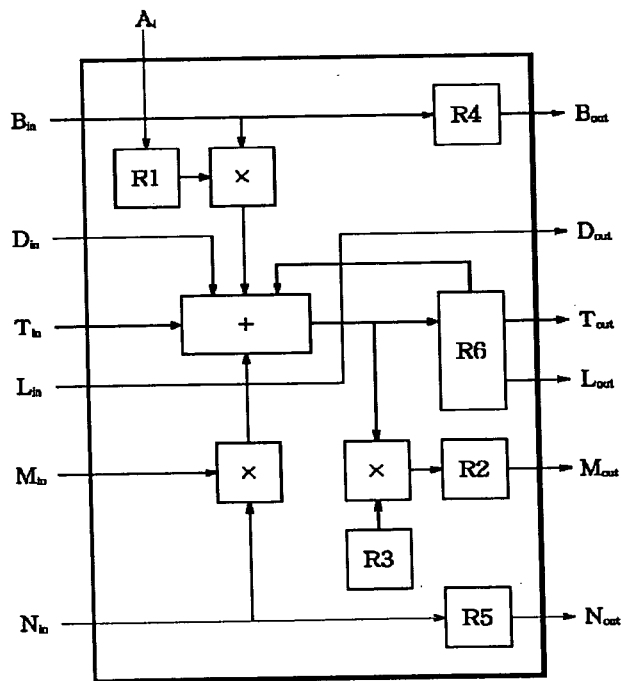
【図9】



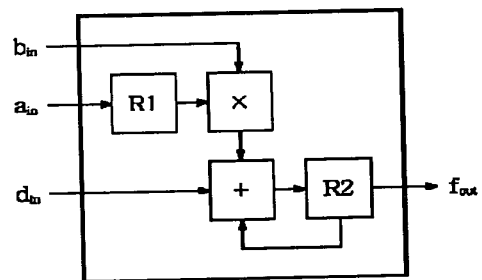
【図4】



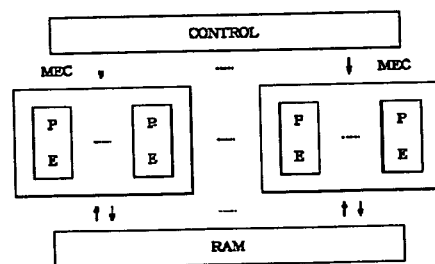
【図5】



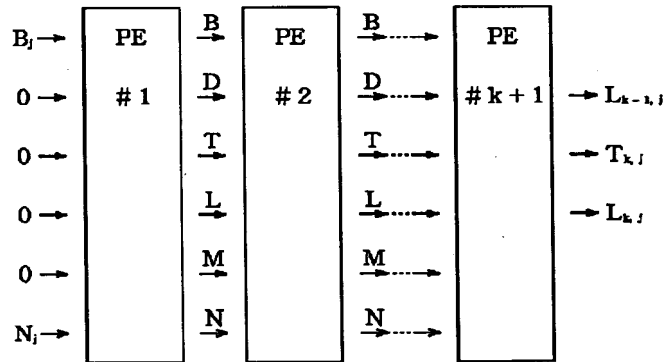
【図16】



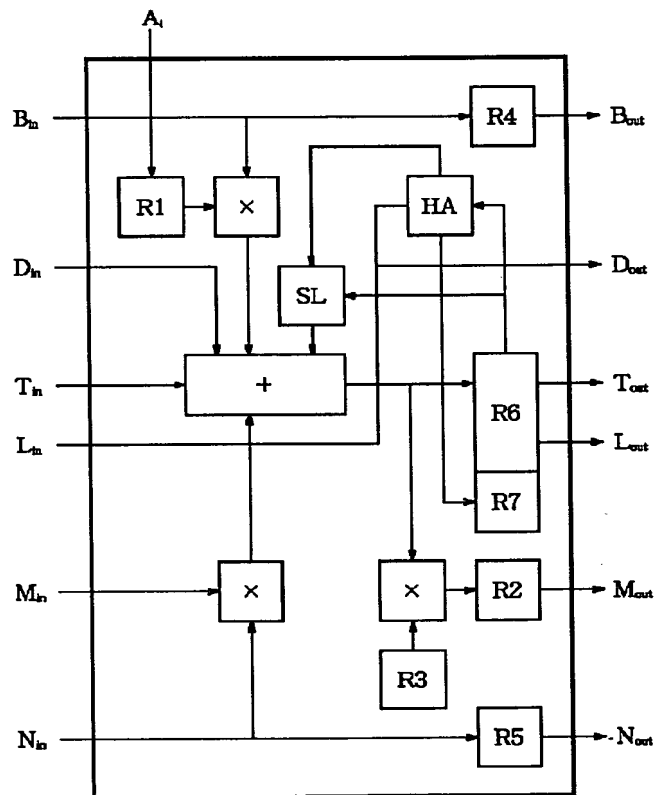
【図19】



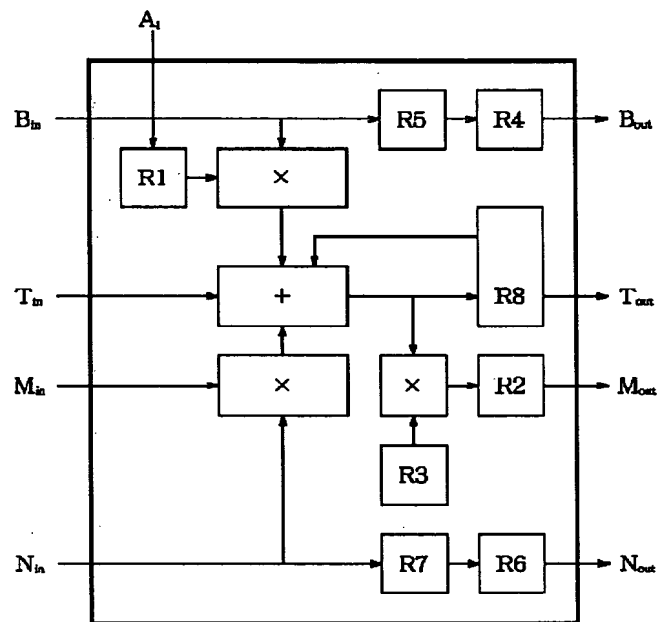
【図6】



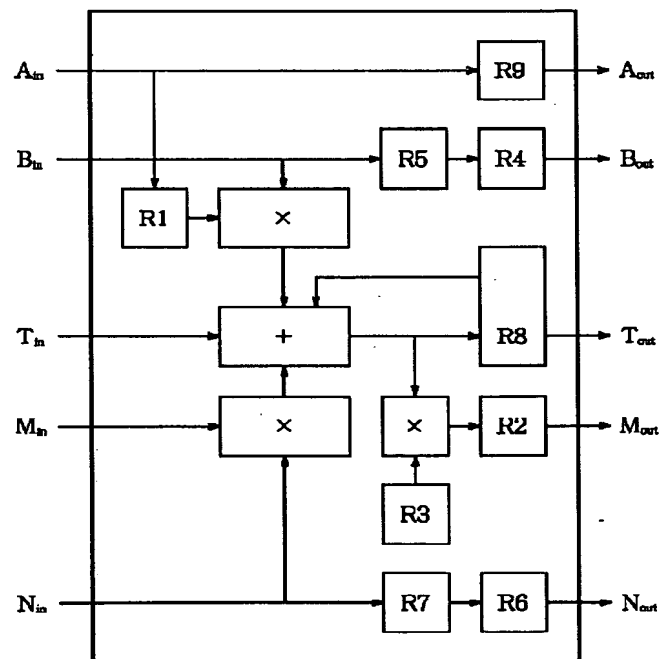
【図7】



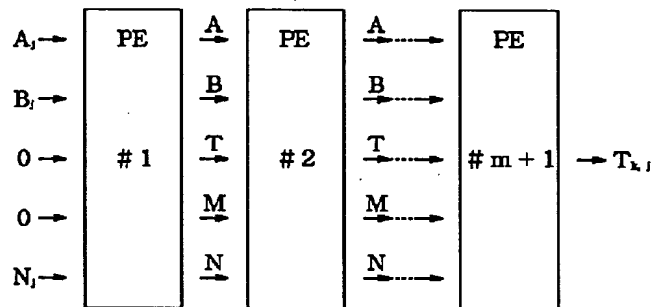
【図8】



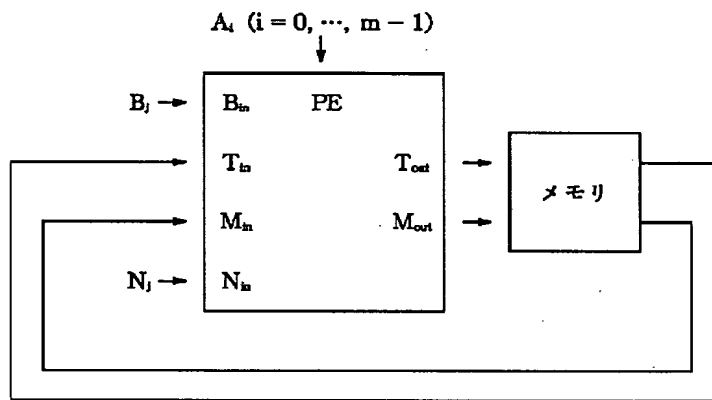
【図10】



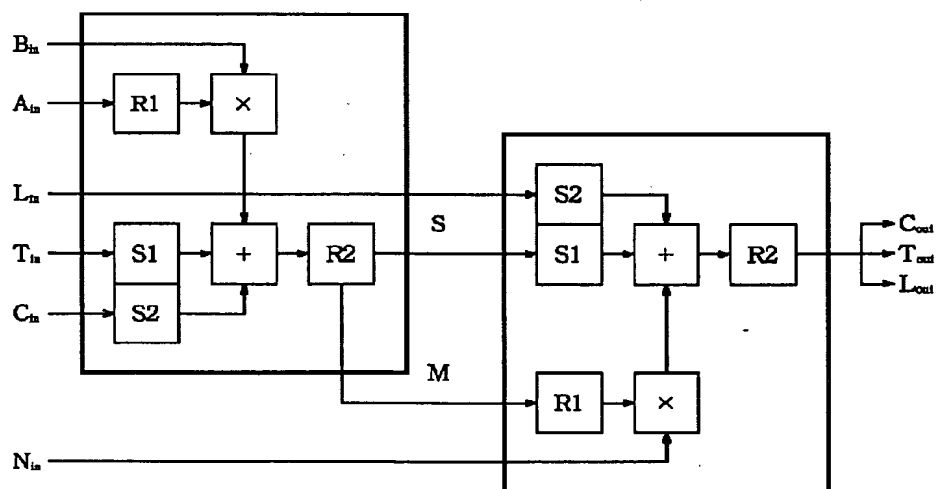
【図11】



【図12】



【図15】



```

graph LR
    A["Ai (i = 0, ..., m - 1)"] --> PEs
    Bi["Bi"] --> PEs
    Ti["Ti"] --> PEs
    Ni["Ni"] --> PEs
    PEs -- Tout --> Mem["メモリ"]
    Mem --> Out[" "]

```